



Projeto Final de Programação INF 2102

Ferramenta de *Reinforcement Learning* para
Controle de Tensão em Sistemas de Potência

Mauricio Raphael Waisblum Barg

Matrícula: 1912718

Orientador: Prof. Marcus Poggi

Rio de Janeiro, 03 de Julho de 2020

Sumário

Introdução.....	3
Objetivo	4
<i>Reinforcement Learning</i>	5
<i>Deep Q-Learning</i>	6
Sistema.....	8
Arquitetura	8
Especificação	9
Casos de Uso.....	10
Criação das Curvas de Carga	12
Configuração do Controle Automático Intrínseco	13
Diagrama de Sequência.....	14
Testes.....	15
Casos de teste	15
Documentação Para o Usuário	17
Instalação de Dependências	17
Escolha dos Arquivos de Circuito	18
Escolha dos Arquivos de Carga	19
Configuração dos Parâmetros.....	19
Execução	20
Análise	20
Conclusão.....	21
Referências	22

Introdução

Este trabalho é um requisito para a disciplina INF2102 – Projeto Final de Programação que busca avaliar a capacidade de se conduzir um processo completo de desenvolvimento de software, desde o planejamento de uma aplicação até seu controle de qualidade. Dessa forma, este documento busca seguir as instruções fornecidas para a disciplina de maneira fiel, atendendo todos os requisitos propostos.

Objetivo

A aplicação descrita neste trabalho tem por objetivo auxiliar a operação em tempo real de sistemas elétricos de potência, mais especificamente o setores de transmissão e distribuição de energia. Este auxílio vem através da indicação e execução das melhores ações a serem tomadas no sistema de forma que sua configuração garanta que o nível de tensão nos diversos pontos do sistema se encontre dentro de limites pré-estabelecidos.

Este processo de manter a tensão do sistema dentro dos níveis adequados é mais comumente chamado de controle de tensão. O controle de tensão é uma atividade que deve ser conduzida pelos operadores constantemente (muitas vezes em um intervalo de minutos) e de forma rápida. Sua não realização pode causar instabilidade e até mesmo desligamentos no sistema, o que vem atrelado a certas penalidades como por exemplo multas e notificações. Devido ao seu caráter de constância e urgência, os operadores não possuem tempo hábil para avaliar qual ação resolverá o problema da melhor forma e ainda avaliar os impactos futuros desta ação, fazendo com que sua escolha seja baseada somente nas experiências similares passadas, sujeitas a “vícios” e tendo um escopo limitado.

Para mitigar este problema, propõe-se então a criação de um agente inteligente baseado em um dos paradigmas de *machine learning* chamado *reinforcement learning*. Mais especificamente, é utilizado o *Deep Q-Learning*, um algoritmo que integra ao *reinforcement learning* a utilização de redes neurais para que o agente possa atuar de maneira mais generalizada. Através da interação com um ambiente (nesse caso o sistema elétrico) o agente é capaz de tomar diversas decisões e avaliar seus impactos, levando em consideração não só as consequências imediatas, mas também o estado futuro do ambiente.

Reinforcement Learning

Reinforcement Learning (RL) ou aprendizado por reforço é juntamente com os aprendizados supervisionado e não supervisionado um dos três paradigmas de *machine learning*. A essência do RL consiste em criar um mapa que associe situações com ações que devem representar a melhor maneira de agir para um determinado cenário. Para isso, um **agente** determina através de tentativa e erro quais ações são as melhores, em um sentido não somente imediato, mas também considerando as consequências futuras desta ação.

De maneira mais técnica, um **agente** toma **ações** que alteram o **estado** de um **ambiente** e através de um **signal de recompensa** consegue avaliar a qualidade da ação tomada. Essas ações são escolhidas seguindo uma **política** e o sinal de recompensa através de um **fator de desconto** é capaz de contemplar também as recompensas futuras. Essa técnica é exemplificada na Figura 1.

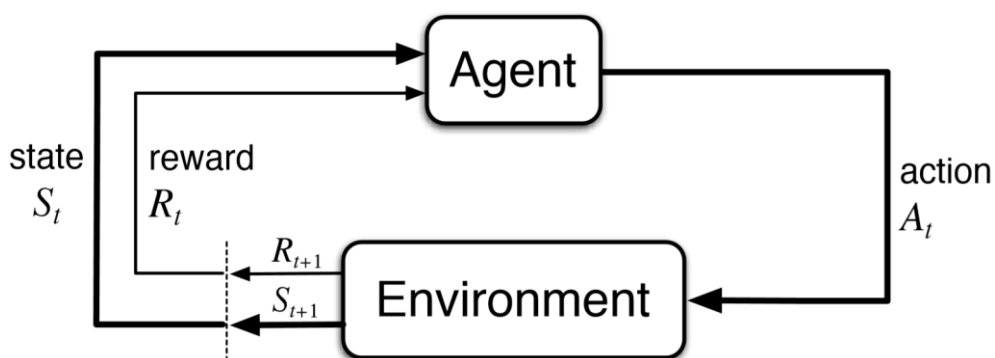


Figura 1 - Diagrama do processo de *Reinforcement Learning*

Este processo pode ser conduzido de formas diferentes, dependendo das características do ambiente e o objetivo esperado. Para o problema de controle de tensão em sistemas de potência, a técnica de RL utilizada foi o *Deep Q-Learning* devido a sua capacidade de lidar com ambientes que possuem um grande ou infinito número de estados possíveis (que é o caso dos sistemas de potência) e por permitir que não se possua um modelo completo do ambiente, onde todas as probabilidades de transição entre os estados são definidas.

Deep Q-Learning

Deep Q-Learning (DQL) é uma das diversas técnicas de RL. O DQL é baseado em uma outra técnica mais simples denominada apenas *Q-Learning* e consiste na combinação desta técnica com redes neurais, para que o agente consiga generalizar o conhecimento adquirido para um grande número de estados. Mais detalhadamente, o Q-Learning cria uma tabela que para cada par (estado, ação) que contém o retorno esperado por se tomar uma determinada ação ao se encontrar em um determinado estado.

Tabela 1 - Tabela utilizada pelo Q-Learning

	Estado 1	Estado 2	Estado 3	...	Estado n
Ação 1	10	-3	1	...	9
Ação 2	2	23	5	...	4
Ação 3	3	-22	10	...	0
Ação 4	11	12	-5	...	1

Para a construção da tabela, são definidos episódios. Episódios são janelas de interação que o algoritmo possui com o ambiente e possuem um número máximo de passos (etapas de um jogo, um intervalo de tempo, etc.). Para cada passo do episódio, uma ação (A) é escolhida da tabela, seguindo alguma política (estratégia). A mudança para de um estado (S') para o próximo (S) é então observada e uma recompensa (R) é recebida. O valor (S, A) da tabela é atualizado seguindo a equação mostrada no algoritmo da Figura 2.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$
Loop for each episode:
 Initialize S
 Loop for each step of episode:
 Choose A from S using policy derived from Q (e.g., ε -greedy)
 Take action A , observe R, S'
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
 $S \leftarrow S'$
 until S is terminal

Figura 2 - Algoritmo Q-Learning

O grande problema em se utilizar o *Q-Learning* é que para a construção da tabela é necessário enumerar todos os estados. Porém, para ambientes onde existem variáveis contínuas (que é o caso de sistemas elétricos), enumerar todos os estados não é factível. É nesse cenário que surge o DQL. Grosso modo, a principal mudança entre o *Q-Learning* e o DQL é a substituição da tabela por uma rede neural, cuja entrada é o estado do sistema e a saída é o retorno esperado para cada ação que pode ser tomada naquele estado. Essa mudança é exemplificada na Figura 3. Além disso, são necessárias ligeiras mudanças para adaptar o treinamento da rede neural e garantir sua convergência. O algoritmo geral é descrito na Figura 4.

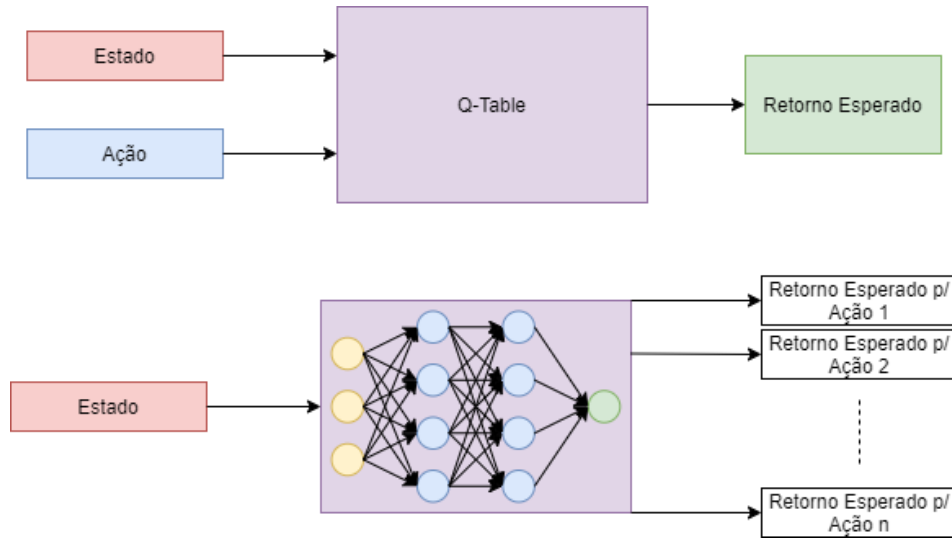


Figura 3 - Comparação *Q-Learning* e DQN

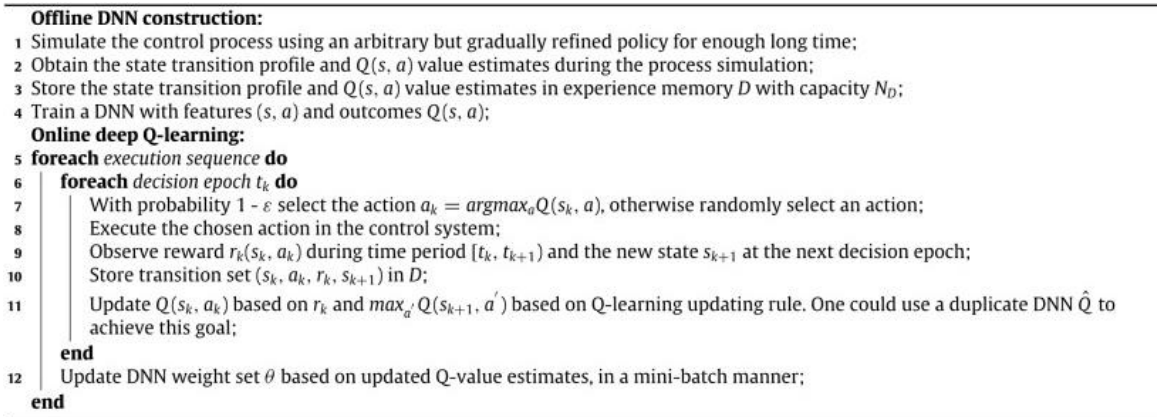


Figura 4 - Algoritmo de DQL

Sistema

Arquitetura

Para desenvolvimento do sistema e algoritmo foi utilizada a linguagem de programação Python versão 3.7.6. No que tange o DQL, os arquivos foram divididos de forma geral em acordo com suas responsabilidades: Agente, Ambiente, Rede Neural e Simulação. Para a criação e treinamento da rede neural foi utilizado o *framework* Tensor Flow.

O módulo de simulação consiste na integração da aplicação com um *software* externo de cálculo de fluxo de potência em sistemas elétricos. Este cálculo é necessário uma vez que é através dele que se torna possível avaliar o nível de tensão em todos os pontos do sistema. Como motor de cálculo foi utilizado o programa OpenDSS, *open source* e desenvolvido pela EPRI (*Electric Power Research Institute*). Este programa fornece as DLLs necessárias para o envio de comandos e obtenção de dados através do Python, sendo necessário somente encapsular estas funcionalidades em funções.

Para a execução do programa, é necessário que o sistema elétrico a ser otimizado pelo RL esteja modelado de acordo com o formato do OpenDSS. Este formato, bem como um guia de como conduzir a modelagem pode ser encontrado na documentação disponibilizada pela EPRI¹.

As bibliotecas necessárias para a execução de todas as funcionalidades do sistema estão listadas no arquivo “requirements.txt” presente na raiz do diretório do programa e devem ser instaladas antes da execução do programa através do comando “pip install -r requirements.txt”. Além disso, é necessário também fazer a instalação do programa OpenDSS, para que as ferramentas de cálculo possam ser utilizadas.

Por fim, para o funcionamento correto do sistema é necessário fornecer para os circuitos exemplos de curvas de carga. Estes arquivos representam a flutuação do consumo de energia ao longo do dia. Os detalhes formato e como construir estes arquivos são apresentados em seções posteriores, na seção de especificação do sistema.

A arquitetura completa do sistema, contemplando todos os arquivos é mostrada no diagrama da Figura 5.

¹ <https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Distrib/Doc/OpenDSSManual.pdf>

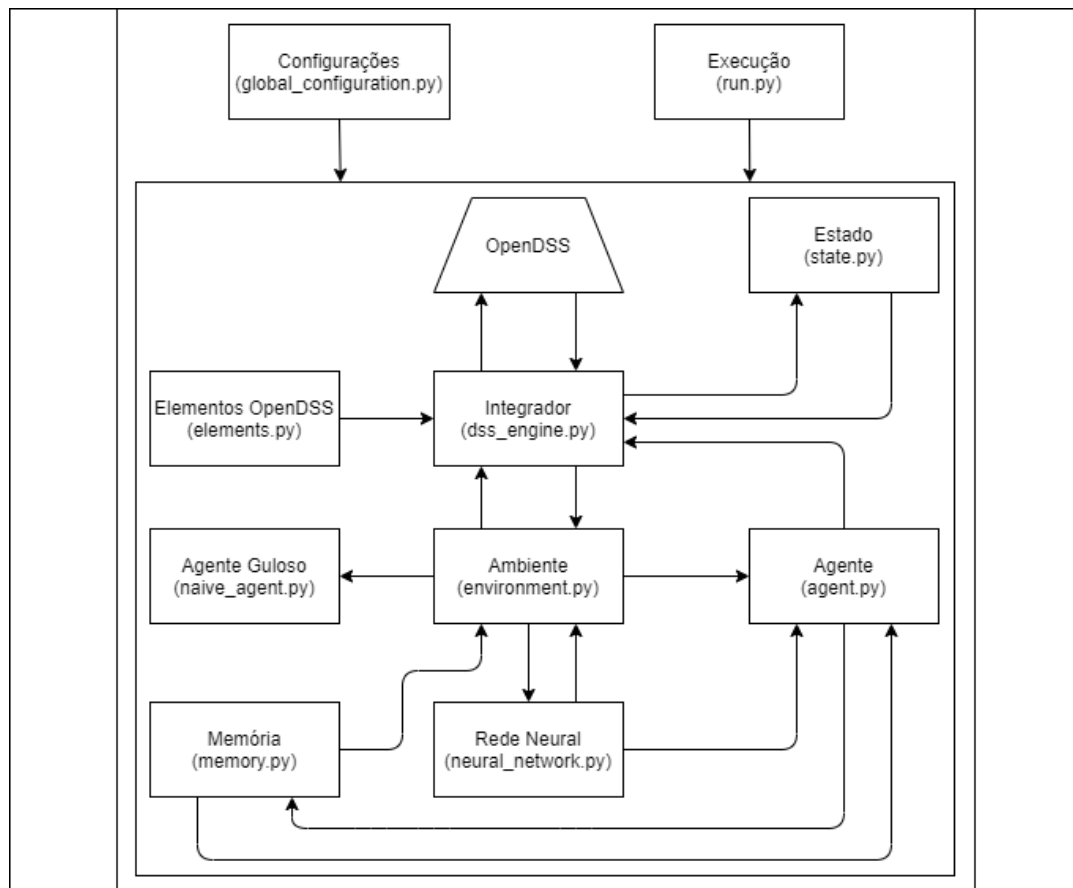


Figura 5 - Arquitetura do Sistema

Especificação

A utilização do sistema se dá em duas etapas distintas. A primeira consiste na escolha do circuito a ser otimizado e treinamento do agente inteligente para este. A segunda etapa, consiste na avaliação dos resultados gerados quando o agente é testado no circuito em que foi treinado, comparando com uma técnica gulosa e com o controle intrínseco presente no sistema. Posteriormente, caso os resultados sejam satisfatórios, é possível implantar o algoritmo e a rede treinada na operação em tempo real.

Por se tratar de uma aplicação com um nível técnico elevado, não existe uma interface amigável para o usuário. Por isso, o usuário deverá ser uma pessoa com conhecimentos no mínimo básicos de Python e um conhecimento intermediário de engenharia elétrica, necessário para a análise dos resultados.

As funcionalidades e personalizações disponíveis para o usuário serão descritas nesta seção através de casos de uso. Também serão mostrados os processos de criação das curvas de carga e de habilitação/desabilitação do controle automático intrínseco dos sistemas, necessários para a execução do algoritmo.

Casos de Uso

Os casos de uso do sistema são mostrados no diagrama da Figura 6. Os detalhes de cada caso como fluxo e requisitos são descritos nas tabelas a seguir.

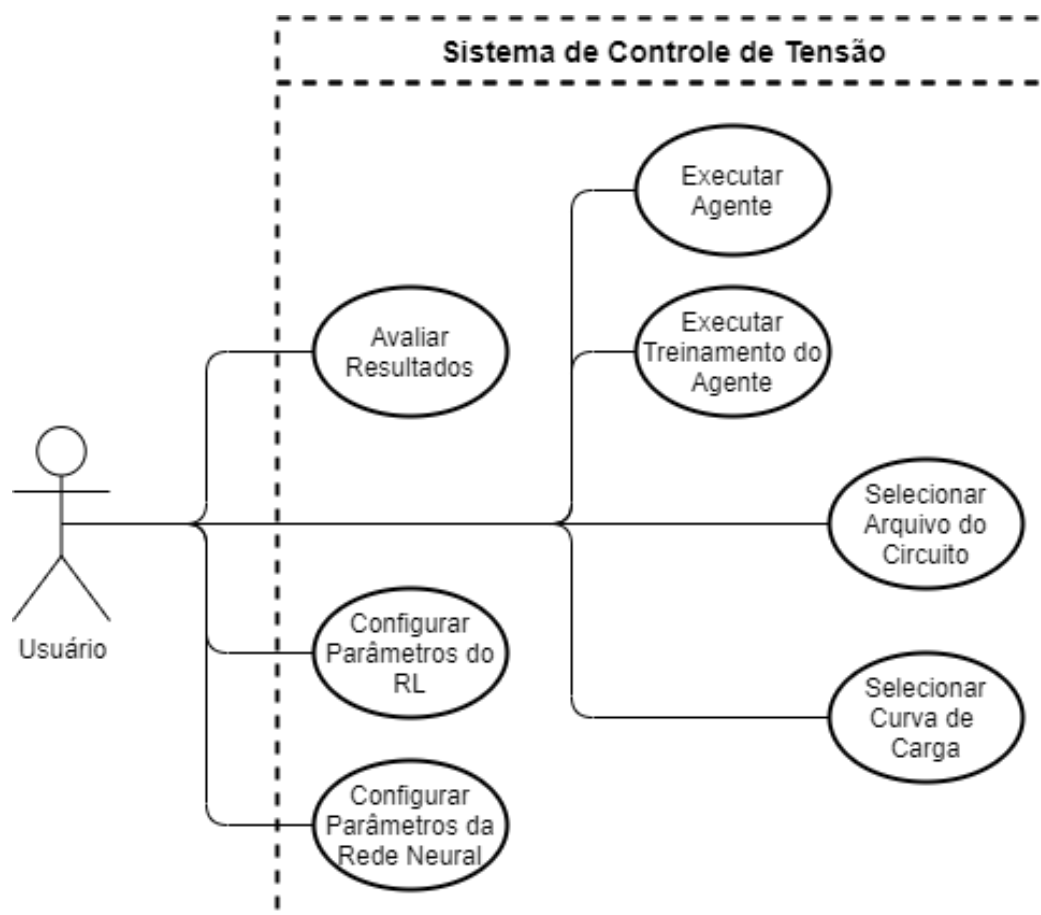


Figura 6 - Diagrama dos Casos de Uso

Nome	Selecionar Arquivo do Circuito
Requisitos	Dependências instaladas
Atores	Usuário
Descrição	O usuário indica para o sistema o caminho do arquivo do circuito a ser utilizado na execução do algoritmo.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “dss_engine.py” para edição; 2. Usuário navega até a função “start”; 3. Usuário insere nas chamadas da função “open_file” o caminho de um arquivo do circuito desejado onde o controle automático do sistema é habilitado e outro onde é desabilitado.

Nome	Selecionar Curvas de Carga
Requisitos	Dependências instaladas e uma curva de carga criada
Atores	Usuário
Descrição	Usuário indica o caminho da pasta contendo um ou mais arquivos de curva de carga.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “dss_engine.py” para edição; 2. Usuário navega até a função “set_load_profile”; 3. Usuário insere no parâmetro “folder” o caminho da pasta contendo um ou mais arquivos de curva de carga.

Nome	Executar o Treinamento do Agente
Requisitos	Dependências instaladas, arquivo selecionado e curvas de carga definidas
Atores	Usuário
Descrição	O usuário inicia o treinamento do agente.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “run.py” para edição; 2. Usuário define o parâmetro “train” como True; 3. Usuário executa o script.

Nome	Executar o Agente
Requisitos	Dependências instaladas, arquivo selecionado, curvas de carga definidas e treinamento realizado
Atores	Usuário
Descrição	Usuário inicia a execução do agente.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “run.py” para edição; 2. Usuário define o parâmetro “train” como False; 3. Usuário executa o script.

Nome	Avaliar os Resultados
Requisitos	Dependências instaladas, arquivo selecionado, curvas de carga definidas e treinamento realizado
Atores	Usuário
Descrição	Usuário avalia os resultados gerados pelo agente treinado.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “run.py” para edição; 2. Usuário define o parâmetro “train” como False; 3. Usuário executa o script; 4. Ao fim da execução, usuário observa os gráficos que aparecerão e compara as técnicas utilizadas.

Nome	Configurar Parâmetros do RL
Requisitos	Dependências instaladas
Atores	Usuário
Descrição	Usuário altera os parâmetros utilizados para o treinamento do agente inteligente.

Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “global_configuration.py” para edição; 2. Usuário altera os parâmetros referentes ao RL e salva o arquivo.
--------------	---

Nome	Configurar Parâmetros da Rede Neural
Requisitos	Dependências instaladas
Atores	Usuário
Descrição	Usuário altera as configurações de arquitetura da rede neural.
Fluxo	<ol style="list-style-type: none"> 1. Usuário abre o arquivo “neural_network.py” para edição; 2. Usuário altera os valores dos parâmetros referentes ao tamanho das camadas da rede neural e salva o arquivo.

Criação das Curvas de Carga

Ao se tratar de sistemas de potência, as curvas de carga representam a flutuação do consumo ao longo do dia. Essa flutuação do consumo, dentre outras consequências se reflete na flutuação da tensão em diferentes pontos do sistema. É justamente este fenômeno que torna necessária a tomada de ações para manter seus valores dentro dos limites adequados. Dessa forma, para simular adequadamente um episódio do *reinforcement learning* (um dia) é necessário alterar a carga do sistema periodicamente de maneira similar a realidade. Para simular este comportamento, são criadas curvas de carga normalizadas, consistindo em fatores que serão multiplicados pela carga original do sistema, simulando a variação ao longo do dia.

Para criar as curvas que serão utilizadas no sistema, primeiro deve-se obter uma curva de um sistema real ou criada através de simulação com no mínimo 1440 pontos e valores de carga em kW (kilowatt). Caso a curva possua mais de 1440 pontos, deve ser feita uma agregação para que contenha exatamente este número de pontos. Em seguida, encontra-se a média das cargas ao longo do dia e divide-se todos os valores de carga por esta média encontrando assim os fatores. Então é criado um arquivo no formato “.csv”, separado com “;” contendo duas colunas. Uma vai de um até 1440 e representa o minuto do dia. A segunda coluna possui o fator associado àquele minuto. Devido à sazonalidade associada aos sistemas elétricos, as curvas muitas vezes são similares nos mesmos dias das semanas. Assim, é importante que se associe um dia da semana à uma curva. Isso é feito através dos três últimos caracteres do nome do arquivo, que devem ser a abreviação dos dias da semana em inglês.

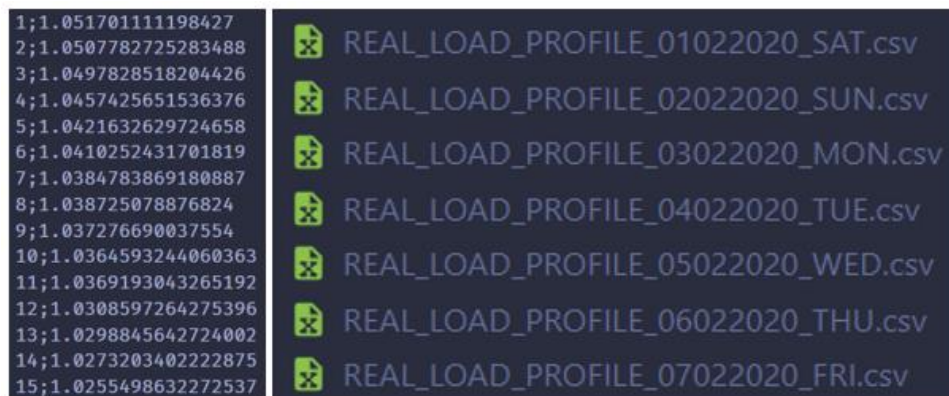


Figura 7 - Formato do arquivo e exemplo dos nomes

É necessário somente um arquivo, uma vez que o sistema toma medidas para inserir “barulhos” na curva e aumentar o número de exemplos. Porém, quanto mais curvas, mais generalizado será o agente treinado.

Configuração do Controle Automático Intrínseco

Nos sistemas de potência, muitas vezes os equipamentos possuem um controle automático que é capaz de chavear e alterar o estado de equipamentos quando a tensão se deteriora. O treinamento do agente pode ser conduzido com esse controle ligado ou desligado. Ainda, é possível comparar a qualidade do controle do agente inteligente (com o controle intrínseco do sistema desligado) com o controle intrínseco do sistema.

Para isso, o sistema necessita apenas que sejam fornecidos dois arquivos do circuito: um com o controle habilitado e outro com o controle desabilitado. Essa alteração deve ser feita no *script* (arquivo “.dss”) do circuito simulado e consiste na inserção do comando “Set Controlmode=OFF” logo antes do comando “Solve”, presente em todos os circuitos.

```
156 Set Controlmode=OFF
157 Solve
```

Figura 8 - Comando para desabilitar o controle intrínseco do sistema

Diagrama de Sequência



Figura 9 - Diagrama de sequência das funcionalidades do sistema

Testes

Para assegurar o correto funcionamento de algumas funções essenciais da aplicação foram criados testes unitários. O arquivo contendo os testes chama-se “tests.py” e se encontra na raiz da aplicação. Para executar os testes, basta executar o script. Abaixo, serão detalhados os casos de teste e seus logs.

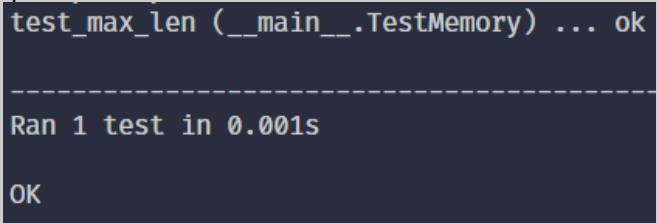
Casos de teste

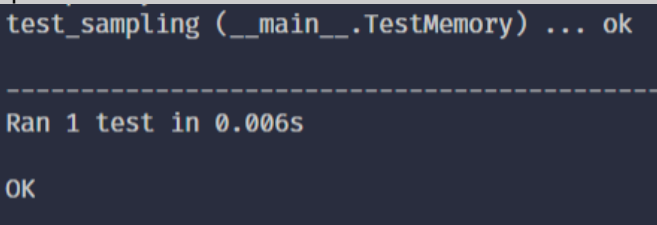
Número	1
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções responsáveis por integrar a aplicação ao OpenDSS.
Caso de Teste	Testar a capacidade de enviar comandos para o OpenDSS.
Procedimento	Executar o script de testes.
Resultado Esperado	O teste passa. <pre>test_engine_connection (__main__.TestEngine) ... 151-B16DCC243796x0x1x0.py Building definitions from type library... Generating... Importing module ok ----- Ran 1 test in 1.092s OK</pre>

Número	2
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções responsáveis por integrar a aplicação ao OpenDSS.
Caso de Teste	Testar a capacidade de abertura de conexão com o OpenDSS.
Procedimento	Executar o script de testes.
Resultado Esperado	O teste passa. <pre>test_open_connection (__main__.TestEngine) ... 1-B16DCC243796x0x1x0.py Building definitions from type library... Generating... Importing module ok ----- Ran 1 test in 1.194s OK</pre>

Número	3
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções responsáveis por integrar a aplicação ao OpenDSS.
Caso de Teste	Testar a capacidade de encerrar a conexão com o OpenDSS.
Procedimento	Executar o script de testes.
Resultado Esperado	<p>O teste passa.</p> <pre> test_kill_connection (__main__.TestEngine) ... 1-B16DCC243796x0x1x0.py Building definitions from type library... Generating... Importing module ok ----- Ran 1 test in 1.005s OK </pre>

Número	4
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções responsáveis por lidar com a utilização da rede neural.
Caso de Teste	Testar a capacidade de fazer cópia de uma rede.
Procedimento	Executar o script de testes.
Resultado Esperado	<p>O teste passa.</p> <pre> test_network_sync (__main__.TestNeuralNetwork) ... -B151-B16DCC243796x0x1x0.py Building definitions from type library... Generating... Importing module 2020-05-21 13:14:42.903187: W tensorflow/stream_ex ; dlerror: nvcuda.dll not found 2020-05-21 13:14:42.906845: E tensorflow/stream_ex 2020-05-21 13:14:42.919228: I tensorflow/stream_ex RDXNTBRJ3033 2020-05-21 13:14:42.936039: I tensorflow/stream_ex 2020-05-21 13:14:42.953724: I tensorflow/core/plat nary was not compiled to use: AVX2 c:/Users/mauricio.barg/Desktop/PFP/app/tests.py:55 self.assertEqual(env._online_network, env._targ ok ----- Ran 1 test in 1.314s OK </pre>

Número	5
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções necessárias para correto funcionamento do RL
Caso de Teste	Testar a capacidade da memória do agente de limitar o número de elementos.
Procedimento	Executar o script de testes.
Resultado Esperado	O teste passa.  <pre>test_max_len (__main__.TestMemory) ... ok ----- Ran 1 test in 0.001s OK</pre>

Número	6
Criticidade	Alta
Localização	tests.py
Objeto de Teste	Funções necessárias para correto funcionamento do RL
Caso de Teste	Testar a capacidade da memória do agente de amostrar aleatoriamente seus elementos.
Procedimento	Executar o script de testes.
Resultado Esperado	O teste passa.  <pre>test_sampling (__main__.TestMemory) ... ok ----- Ran 1 test in 0.006s OK</pre>

Documentação Para o Usuário

Nesta seção serão descritos os procedimentos que devem ser tomados para a correta utilização da ferramenta.

Instalação de Dependências

O primeiro passo para a execução a execução do sistema é a instalação de todas as dependências. Para fazer isso, primeiro deve-se criar um “*virtualenv*” do Python. Para isso digite na linha de comando, aberta na pasta raiz do programa o seguinte comando: “python -m venv env”. Então, para ativar o “*virtualenv*” deve-se navegar até a pasta “env”, então “*scripts*” e executar o comando “*activate*”.

```
(base) C:\Users\mauricio.barg\Desktop\PPF\app>pip install -r requirements.txt
```

Figura 10 - Comando de instalação das dependências

Em seguida deve-se executar o comando “pip install -r requirements.txt” na linha de comando na pasta raiz da aplicação.

```
(base) C:\Users\mauricio.barg\Desktop\PPF\app>python -m venv env
```

Figura 11 - Comando de criação do "virtualenv"

A próxima etapa é a instalação do OpenDSS². Ao baixar o arquivo basta seguir as instruções presentes no instalador.

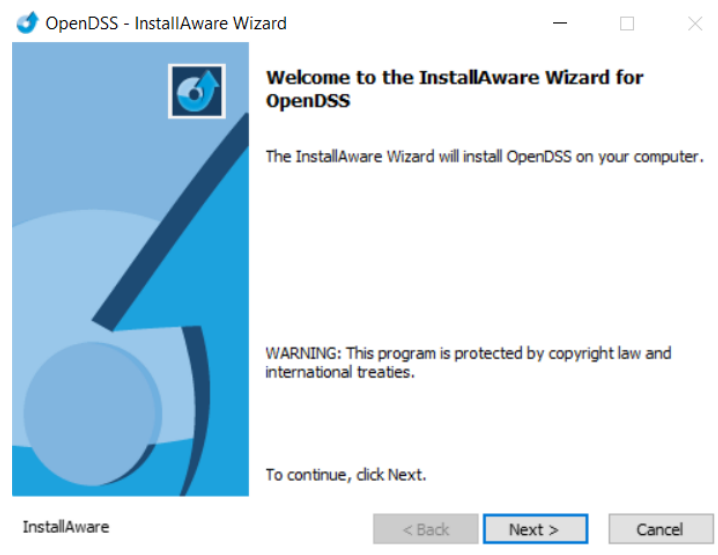


Figura 12 - Tela de instalação do OpenDSS

Escolha dos Arquivos de Circuito

Para definir os arquivos deve-se ir até o arquivo “dss_engine.py” e navegar até a função “start”. Os caminhos dos arquivos (definidos na seção anterior) devem ser inseridos nas funções “open_file” conforme mostrado na Figura 13.

² <https://sourceforge.net/projects/electricdss/>

```
def start(self, tap_changes=False):
    """
    Executa uma série de funções necessárias para iniciar a simulação.

    Parâmetros:
    | tap_changes (bool): Habilita ou não o controle automático inerente do circuito.

    Erros:
    | None

    Retorna:
    | None

    """
    if tap_changes:
        self.open_file()
    else:
        self.open_file()
```

Figura 13 - Local de inserção dos caminhos dos arquivos dos circuitos utilizados na simulação

Escolha dos Arquivos de Carga

Para definir a pasta contendo os arquivos da curva de carga deve-se ir até o arquivo “dss_engine.py” e navegar até a função “set_load_profile”. O caminho da pasta deve ser inserido no parâmetro “folder” da função conforme mostrado na Figura 13.

```
def set_load_profile(self, folder=''):
    """
    Carrega o perfil de carga.
    O formato do arquivo é: id_da_amostra;carga_normalizada. ex. 1;1.0323, 2;1.00 ...
    Os últimos três caracteres do nome do arquivo devem ser os dias da semana em inglês: MON, TUE, ...
    Existe um processo aleatório que insere distúrbios com uma certa probabilidade, baseado em "Perlin Noise".

    Parâmetros:
    | folder (str): Caminho da pasta onde os arquivos das curvas de carga estão;

    Erros:
    | Excessão caso não haja arquivos.

    Retorna:
    | None

    """
```

Figura 14 - Configuração dos arquivos de carga

Configuração dos Parâmetros

Para configuração dos diversos parâmetros utilizados no sistema, deve-se abrir o arquivo “global_configuration.py” e editar os parâmetros conforme mostrado na Figura 15.

```

# Rewards and Penalties
self.__OUT_OF_LIMITS_PENALTY = -1
self.__TOWARD_TARGET_REWARD = 0.5
self.__AWAY_TARGET_PENALTY_OVER = -0.95
self.__AWAY_TARGET_PENALTY_UNDER = -0.75
self.__STAND_STILL_REWARD_OUT_OF_TARGET = 0.1
self.__STAND_STILL_REWARD_ON_TARGET = 0.9
self.__STAND_STILL_PENALTY = -0.5
self.__MEANINGLESS_ACTION_PENALTY = -0.8
self.__ACTION_UNDONE_PENALTY = -1

# DQL configuration
self.__DISCOUNT_FACTOR = 0.9
self.__BASE_LEARNING_RATE = 0.001
self.__MAX_EPSILON = 1
self.__MIN_EPSILON = 0.2
self.__MEMORY_BATCH_SIZE = 512
self.__TARGET_UPDATE_FREQUENCY = 2048
self.__REPLAY_START = 512
self.__MAX_MEMORY_SIZE = 8192

```

Figura 15 - Parâmetros configuráveis do sistema

Execução

Para executar a aplicação deve-se abrir o arquivo “run.py” e definir se o agente será treinado ou somente executado. Destaca-se que para executar o agente é necessário treiná-lo pelo menos uma vez.

```

train = True
env = Environment()
if train:
    env.train()
else:
    env.run()

```

Figura 16 - Arquivo de execução da aplicação

Análise

Ao final da execução, diversos gráficos serão exibidos e o usuário poderá avaliar a qualidade da solução através deles. Um exemplo é mostrado na Figura 17.

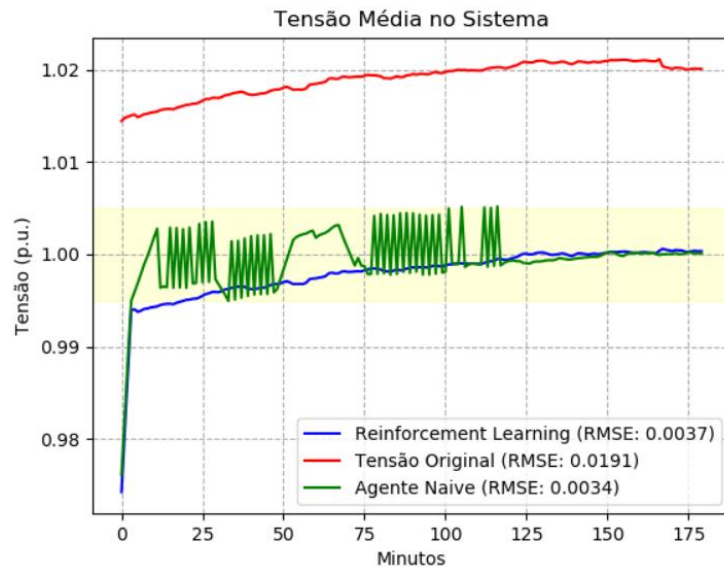


Figura 17 - Exemplo de resultado exibido ao final da execução

Conclusão

O sistema foi documentado seguindo as regras disponíveis no roteiro da disciplina. Foram conduzidos testes unitários para assegurar o funcionamento das principais funcionalidades do programa. Ademais, o código fonte foi comentado seguindo-se as indicações presentes na documentação da linguagem Python.

Referências

R. Sutton and A. Barto, *Reinforcement learning*. Cambridge, Mass: The MIT Press, 2018.